META DATA FRAMEWORK FOR ENTERPRISE INFORMATION SYSTEMS SPECIFICATION

Aiming to reduce or remove the development phase for EIS systems

Jon Davis, Andrew Tierney, Elizabeth Chang
School of Information Systems, Curtin University of Technology, GPO Box U1987,Perth WA 6845, Australia

Keywords: Enterprise Information System, meta-data, framework, meta-data driven application, application generation

Abstract: This paper proposes a process for implementing a meta-data approach to defining a platform independent operational computer system application. It identifies Enterprise Information System (EIS) type systems as ideal candidates for implementation using this meta-data, based on the simplification opportunities available due to the typically visual and transactional component bias of EIS systems. It describes architecture for the development of a suitable meta-data based application generator system. This development could lead to the generation of new accelerated EIS development methodologies in business modeling, analysis, design, system deployment and global information exchange.

1 INTRODUCTION

Common core system development methodologies such as the Waterfall, Spiral, Fountain and V models (Sommerville, 2000), (IEEE, 1998), (IEEE, 1997) have not been fundamentally altered as a result of modern technologies and in general we are still maintaining the basic paradigm for system development; analysis, design, develop code, test and deploy. New system development methodologies such as Prototyping, Agile Processes, Big Ball of Mud (Pressman, 2000), (Martin, Raffo, 2000), (Donzelli, Iazeolla, 2001), typically propose differing levels of task decompositions, parallelism, customer interaction etc and certainly do provide specific advantages when dutifully employed but they are not guaranteed to necessarily change the magnitude of the total effort.

An Enterprise Information System (EIS) must necessarily start with a review of its requirements and the preparation of a design. Our project proposes that performance of the analysis combined with an efficient collection of this information can also perform the bulk of the design phase, largely as a simultaneous activity. Hence the two steps may be merged in our proposed model.

Further, with the collective design requirements stored and available in a suitable meta-data format (from the first stage), we believe that most (if not all) EIS systems can be executed automatically with the availability of suitable runtime components. This expectation is based on the well structured nature of EIS applications; highly visual and interactive applications that prompt for the entry of appropriate data by the users, employ strong rule based actions and utilize database transactions to complete the action.

The introduction of such an approach has the potential to drastically reduce the time to develop and deploy an EIS system. Effectively, once the analysis and design have been completed the system would become available for immediate use! The virtual elimination of the coding combined with the minimisation of the testing and deployment stages has significant benefits for both the developer and the end users.

This project aims to develop an alternative development methodology using a meta-data standard that can be extended upon for defining and producing Enterprise Information Systems.

2 PROGRESSIVE MODELING METHODOLOGIES

This project will be influenced by feature subsets of these improving and emerging technologies:

- **4th Generation Language:** The most common 4GL standard is Structured Query Language (SQL) which has become the standard in

Davis J., Tierney A. and Chang E. (2004).
META DATA FRAMEWORK FOR ENTERPRISE INFORMATION SYSTEMS SPECIFICATION - Aiming to reduce or remove the development phase for EIS systems.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 451-456
DOI: 10.5220/0002626404510456

database management, and will be supported as the data repository standard for this project. Mainstream progress in application development was restricted to proprietary tools although UML (OMG, 2003), (Kruchten, 2000), (Quatrani, 2001) has emerged as a popular and quasi-standard design toolset.

- **Unified Modeling Language**: UML has been progressively developed into a precise toolset for the specification of system requirements, and is being provided with continually improving third party code generation options (Lethbridge, Laganiere, 2002), (Allegrini, 2002), (Guizzardi, Herre, Wagner, 2002). A weakness of UML is that it has such extraordinary breadth and requires correspondingly high technical skills, requiring the difficult marriage of UML experts with business analysts and business process experts to obtain success in the commercial world. We believe that a less complex specification can be achieved for the subset of issues in EIS systems (and that can subsequently be used with far less technical expertise) and will draw on UML features.
- **Object Modeling Group - Model Driven Architecture**: OMG/MDA (OMG, 2003), (Tolk, 2002) provides a solid guiding methodology based on the use of UML to provide the platform independent model, and following through with the efficient development and deployment of the target system. This project will utilize the general philosophy of MDA combined with subsets of modified UML.
- **DARPA Agent Markup Language and Ontology Inference Layer**: DAML+OIL (DARPA, 2003), (W3C, 2001), (Qasem, 2001), (van Harmelen, Horrocks, 2000) provides a foundation for the classification of elements and as a means for automating inference in data sets and has been inspired by the burgeoning information base available in and via the Internet. DAML+OIL lend useful schema elements and provide a logical documentation and information interchange framework.
- **OWL Web Ontology Language**: OWL (W3C, 2004), (Kendall, Dutra, 2002) is a web ontology language derived from DAML+OIL which is seeking candidate recommendation with W3C. The EIS ontology that is ultimately finalized by this project would sensibly seek to utilize OWL.
- **Interface Definition Language**: IDL (IBM, 2003) is the OMG's language for the development of programming language independent object interfaces. Its relevance for this project is as a specification source for modeling generic objects.

These models tend to be concerned with providing methodologies for system analysis and design. This research aims to create an automated solution for EIS systems development from a meta-data framework. It utilizes a meta model driven architecture and ontology based computing technologies to provide the target platform independent meta-data model of an entire EIS specification, along with prototype platform dependent runtime components.

3 CONCEPTUAL FRAMEWORK FOR META-DATA APPROACH

The success of a meta-data approach will be largely reliant on the functionality that can be provided as defined by higher level components with the associated efficiencies gained by the corresponding reduction in continual individual duplication of components. This approach is not exclusive of the need for access to lower level components but for EIS systems it is the ready availability of the higher level functions that will provide the expected major efficiencies.

An inherent feature of EIS systems is the highly visual and interactive nature of the applications. The success of EIS applications is reliant on the entry of appropriate data by the application users, and they are heavily biased towards rules based responses and database transactions as the appropriate action.

A common paradigm in modern code generation that has become a virtual standard following on from the advent of event driven programming is a separation of the actions (implemented as events) from an underlying framework that provides a structure that makes logical sense for the individual applications.

The model depicted in Figure 1 provides for a separation of the structure and events, extending both as managed hierarchies and allowing for flow control between the structures. The model borrows heavily from the use of visual components to provide application structure - this is a commonplace analogy used in most modern Integrated Development Environments (IDE) for the code based generation of applications and from other research works in meta modeling (Motik, Maedche, 2002), (Celms, Kalnins, Lace, 2003).

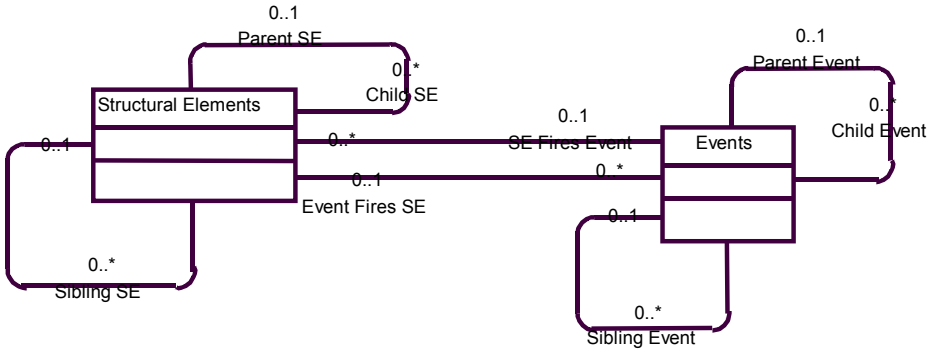

Figure 1. Structural Element - Event Molecule

Basic class diagram indicates a similar hierachical structure as described in the atomic model, for the Structural Elements and Events, with the addition of the relationships between them that will provide the logical basis of their mutual behaviour and interoperability.

4 META-DATA BASED APPLICATION GENERATOR

We have proposed a prototype of a meta-data driven system for the development of EIS systems that aids the specification of the target application, followed by the conversion of that specification into the appropriate meta-data that will then be interpreted by the runtime components. Figure 2 presents an overview of meta-data based application architecture.

4.1 Meta-Data Definer

The meta-data definitions effectively become the application thus it is crucial that efficient methods of defining the meta-data are available.

As the meta-data represents the output of the system specification process then minimal opportunities exist to expedite the original human business analysis effort. However substantial shortcuts will be offered by reductions in the system design effort due to the amount of application infrastructure that would necessarily be provided by the runtime engine in support of the higher level components that the meta-data syntax is modeled upon. This automatic execution of platform independent models to platform dependent execution is a source of major savings in the development cycle.

When starting a new application system design a comprehensive design editor is a requirement for the efficient specification and entry of the system design. A custom editor is required that would represent a suitable design paradigm which is necessarily biased towards the production of the target meta-data syntax.

The field of system analysis and design has developed considerable expertise in providing CASE toolsets to reduce the overall system development effort, particularly the introduction of UML based tools. Opportunities will become available to develop utilities that import design models from existing comprehensive third party design toolsets and convert these designs into the corresponding meta-data syntax.

The database is an integral component of EIS systems which have a strong reliance on data dictionaries and rules based database transactions. Database schemas that do not obfuscate the original design by abstracting the names or types of schema elements, or by relocating database constraints or stored procedures to a remote system tier have significant potential as a starting point to the re-engineering of an existing system.

The development of utilities to reverse engineer existing database schemas and convert the schema to the corresponding data dictionary meta-data can accelerate the meta-data system design process appreciably. Well formed database schemas (without obfuscation) that seek to fully utilize the validation options of the modern RDBMS (Relational Database Management Systems) have the potential to reverse engineer directly to a fully working meta-data based application, requiring optimally minimal or no modification to the meta-data.

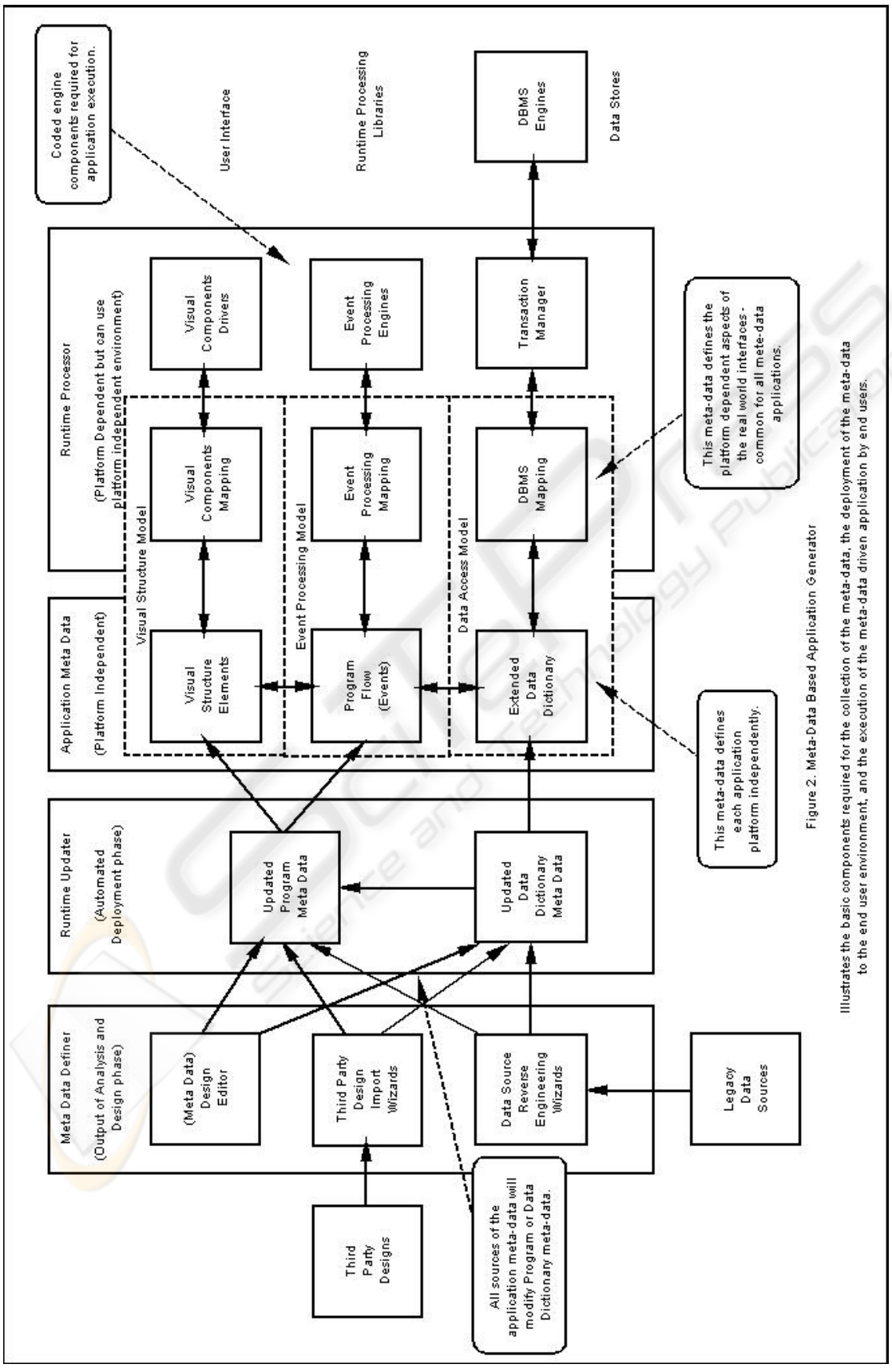

Figure 2. Meta-Data Based Application Generator

Illustrates the basic components required for the collection of the meta-data, the deployment of the meta-data to the end user environment, and the execution of the meta-data driven application by end users.

4.2 Runtime Meta-Data Updater

The magnitude of the effort expended on future system maintenance for systems represents by far the majority effort for the lifecycle of systems from the developer's perspective.

Contributing causes to the ongoing high maintenance costs continue to be the:

- lack of consistent and available system documentation,
- inconsistently applied standards during different (re)development phases,
- lack of structured programming techniques,
- extension of the system to provide features that would have been better served by a full or partial redevelopment,
- natural attrition of software development team members knowledgeable of the system architecture,
- natural progression of the underlying technology to newer, richer and better supported platforms.

The indicated high level maintenance figures of 80% and higher refers to the costs for the developers of the system. For EIS systems this represents person years of effort for the developer. The use of meta-data based applications will drastically reduce these costs to the developer and user base.

For the developer, meta-data applications will largely be self documenting (except for the runtime engine) which reduces the risk and reliance on individual developers and development management practices.

For the customers, meta-data applications are provided as streams of structured meta-data. Unless provided with an updated runtime engine (an occasional requirement), the same runtime engine as deployed to all users at a site would remain unchanged (although this is a relatively simple change management scenario to resolve). Changes to a meta-data application are simply a new stream of meta-data that represent only the specific application changes. To progress to any later version of a meta-data application requires only the application of the correct sequence of progressive update version meta-data streams.

4.3 Application Meta-Data

The ultimate aims of a meta-data model for EIS systems definition include abstraction from the physical constraints of any runtime components that are used for the final implementation and execution by the users.

Utilising a meta-data based interpretation of the application specification allows applications to be executed using any simultaneous combination of platforms that are supported by the components of the runtime engine, providing a progression towards complete platform independence.

The visual structure meta-data is used to construct the appearance of the application as presented by the user interface runtime components to the users. Visual structure meta-data is analogous to the drag'n'drop forms functionality provided by modern GUI based IDE software and will draw on advances such as XForms (W3C, 2004), (Grimes, 2002).

The program flow meta-data is used to define the user interface and local platform actions and procedures that are executed in response to user actions and other data changes. Program flow meta-data is analogous to the event processing and general programming functionality provided by modern GUI based IDE and traditional programming languages.

The data dictionary meta-data is used to define the requirements of the database schema and the data changes required in response to user actions and other data changes. Data dictionary meta-data is analogous to the data dictionary role provided by modern RDBMS systems.

4.4 Runtime Processor

The application meta-data must be interpreted and executed for the users. This role is provided by additional meta-data to map the generic application meta-data to the interface requirements of the platform specific components for execution, plus the platform specific runtime engines that will perform the execution.

5 RECAPITULATION AND FUTURE WORKS

In this paper we proposed a development and deployment of Enterprise Information Systems that uses meta-data application definitions within a framework of the OMG/MDA and the prototype of the meta-data application generator that we believe will reduce the complexity and costs at each phase of the software life cycle. We also propose to develop a meta-data standard for the definition of EIS type systems that will be capable of extension in the future to model a wider spectrum of application types. We presented an overview of the essential components of meta-data application generator architecture.

Currently the prototype system incorporates components of meta-data application architecture including:

- a meta-data syntax for the structure of EIS applications and the visual presentation of the user interface,
- a meta-data syntax for the execution of distributed events,
- a meta-data syntax for the data schemas of EIS applications and conversion between third party database schema types,
- runtime processing components suitable for executing the EIS meta-data on a selected platform subset.

We also note this perspective of the EIS software life cycle could lead to many new concerns in the next generation of software engineering:

- New cross platform components that can broaden the accessibility of meta-data applications,
- Extended meta-data definitions and functionality for richer application types,
- Merging of meta-data applications to create broader and deeper EIS applications,
- Development of more accessible design toolsets.

REFERENCES

Sommerville, I. (2000). *Software Engineering (6th Edition)*, Addison-Wesley Pub Co.

IEEE (1998). *IEEE Standards Collection Software Engineering 1993, 1998 Ed*, IEEE Inc.

IEEE (1997). *IEEE Guide for Developing Software Life Cycle Processes*, IEEE Inc.

Pressman, R. (2001). *Software Engineering: A Practitioner's Approach, 5th Ed*, McGraw-Hill Inc.

Martin, Robert H., Raffo, David (2000). A model of the software development process using both continuous and discrete models in *Proceedings of Software Process: Improvement and Practice, Volume 5, Number 2-3, June-September 2000*.

Donzelli, Paolo, Iazeolla, Giuseppe (2001). A hybrid software process simulation model in *Proceedings of Software Process: Improvement and Practice, Volume 6, Number 2, June 2001*.

OMG (2003). *Introduction to OMG's Unified Modeling Language*. Retrieved October, 10 2003 from http://www.omg.org/gettingstarted/what_is_uml.htm.

Kruchten, Philippe (2000). *The Rational Unified Process: An Introduction*, Addison-Wesley Pub Co.

Quatrani, Terry (2001). *Introduction to the Unified Modeling Language*, Rational User Conference, 2001.

Lethbridge, Timothy, Laganiere, Robert (2002). *Object-Oriented Software Engineering: Practical Software Development using UML and Java*, McGraw-Hill.

Allegrini, Tiziana (2002). *Code Generation Starting From Statecharts Specified in UML*. Universita Degli Studi di Pisa.

Guizzardi, Giancarlo, Herre, Heinrich, Wagner, Gerd (2002). Towards Ontological Foundations for UML Conceptual Models in *Proceedings of 1st International Conference on Ontologies, Databases and Applications of Semantics, 2002*.

OMG (2003). *OMG Model Driven Architecture – The Architecture of Choice for a Changing World*. Retrieved October, 13 2003 from http://www.omg.org/mda/executive_overview.htm.

Tolk, Andreas (2002). *Avoiding Another Green Elephant – A Proposal for the Next Generation HLA Based on the Model Driven Architecture*, 2002 Fall Simulation Interoperability Workshop.

DARPA (2003). *Why Use DAML ?* Retrieved September 12, 2003 from http://www.daml.org/2002/04/why.html.

W3C (2001*). DAML+OIL (March 2001) Reference Description*. Retrieved September, 17 2003 from http://www.w3.org/TR/daml+oil-reference.

Qasem, Abir (2001). *A Prototype DAML+OIL Ontology IDE*, International Semantic Web Working Symposium, Stanford, 2001.

van Harmelen, Frank, Horrocks, Ian (2000). *Questions and Answers on OIL: the Ontology Inference Layer for the Semantic Web*. IEEE Intelligent Systems, Volume 15, Number 6, December 2000.

W3C (2004). *OWL Web Ontology Language Reference*. Retrieved February, 10 2004 from http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

Kendall, Elisa F., Dutra, Mark E. (2002). *An Introduction and UML Profile for the Web Ontology Language (OWL)*, Sandpiper Software Inc, 2002.

IBM (2003). *Interface Definition Language (IDL)*. Retrieved September, 25 2003 from http://www-106.ibm.com/developerworks/websphere/WASInfoCenter/infocenter/wasee_content/corbaio/ref/rcidl.htm.

Motik, Boris, Maedche, Alexander, Volz, Raphael (2002). A Conceptual Modeling Approach for Semantics-Driven Enterprise Applications in *Proceedings of 1st International Conference on Ontologies, Databases and Applications of Semantics, 2002*.

Celms, Edgars, Kalnins, Audris, Lace, Lelde (2003). *Diagram Definition Facilities Based on Meta-Model Mappings*, The Third OOPSLA Workshop on Domain Specific Modeling, 2003.

W3C (2004*). Xforms – The Next Generation of Web Forms*. Retrieved February, 16 2004 from http://www.w3.org/MarkUp/Forms/.

Grimes, Richard (2002). *Developing Applications with Visual Studio .NET*, Addison-Wesley Pub Co.